Controlling a camera in a virtual environment.

E. Marchand, Nicolas Courty

The Visual Computer

Controlling a camera in a virtual environment

É. Marchand, N. Courty

IRISA – INRIA Rennes,
Campus universitaire de Beaulieu,
35042 Rennes Cedex, France
E-mail: {marchand,ncourty}@irisa.fr

Published online: 24 January 2002

This paper presents an original solution to the camera control problem in a virtual environment. Our objective is to present a general framework that allows the automatic control of a camera in a dynamic environment. The proposed method is based on the *image-based control* or visual servoing approach. It consists of positioning a camera according to the information perceived in the image. This is thus a very intuitive approach of animation. To be able to react automatically to modifications of the environment, we also considered the introduction of constraints into the control. This approach is thus adapted to highly reactive contexts (virtual reality, video games). Numerous examples dealing with classic problems in animation are considered within this framework and presented in this paper.

Key words: Automatic camera motion – Automatic cinematography – Visual servoing – Animation

Correspondence to: Éric Marchand

1 Overview

1.1 Issues

There are numerous issues related to the control of a camera in a virtual environment. Typically, the control of the camera is handled by lookat/lookfrom techniques associated with the definition of 3D trajectories. The camera must, usually, first position itself with respect to its environment and must then react in an appropriate and efficient way to modifications of the environment. As regards the first issue, even if full knowledge of the scene is available, as in the computer animation context, the positioning task is not a trivial problem (see [2]). There is a need for precise control of the 6 degrees of freedom (d.o.f.) of the camera in 3D space. The second issue, which can be defined as the introduction of constraints to the camera trajectory, is even more complex. In order to be able to consider unknown or dynamic environments and to achieve real-time camera motion control, these constraints must be properly modeled and "added" to the positioning task.

1.2 Related work

Visual servoing has proved, within the robotics context, to be an efficient solution to these problems. Visual servoing or image-based camera control consists of specifying a task (mainly positioning or target tracking tasks) as the regulation in the image of a set of visual features [6, 8, 21]. A set of constraints is defined in the image space (e.g., "I want to see the tree vertical and centered in the image, while the head of the man must appear in the upper left part of the image"). A control law that minimizes the error between the current and desired positions of these visual features can then be automatically built. A good review and introduction to visual servoing can be found in [10]. As the task specification is carried out in 2D space, it does not require a 3D relationship between objects. However, since the approach is local, it is not a priori possible to consider planning issues. If the control law computes a motion that leads the camera to undesired configurations (such as occlusions or obstacles), visual servoing fails. Control laws taking into account these "bad" configurations therefore have to be considered. Framework that allows the consideration of such constraints has been presented in, for example, [15, 16]. It combines the regulation of the vision-based task with the minimization of cost functions reflecting the constraints imposed on the trajectory.

The Visual Computer (2002) 18:1–19
Digital Object Identifier (DOI) 10.1007/s003710100122

Viewpoint control has also received attention in computer graphics. The main difference with respect to computer vision or robotics is that the problem is no longer ill-posed. Indeed, in this case full knowledge of the scene is available. Even in an interactive context, the past and current behavior of all the objects is fully known. Ware and Osborn [20] have considered various metaphors to describe a six-d.o.f. camera control, including "*eye in hand*". Within this context, the goal was usually to determine the position of the "eye" with respect to its six d.o.f. in order to see an object or a set of objects at given locations on the screen. User interfaces such as a 3D mouse or a six d.o.f. joystick could be considered to control such a virtual device. Obtaining smooth camera motions requires a skilled operator and has proven to be a difficult task. The classical lookat/lookfrom/vup parameterization is a simple way to achieve a gazing task on a world-space point. However specifying a complex visual task within the lookat/lookfrom framework is quite hopeless. Attempts to consider this kind of problem have been made by Blinn [2]; however, the proposed solutions appear to be dedicated to specific problems and hardly scaled to more complex tasks. Image-based control has been described within the computer graphics context by Gleicher and Witkin [7], who called it "*through-the-lens camera control*". They proposed the achievement of very simple tasks such as positioning a camera with respect to objects defined by static "virtual" points. This technique, very similar to the visual servoing framework, considers a local inversion of the nonlinear perspective viewing transformation. A constraint optimization is used to compute the camera velocity from the desired motion of the virtual point in the image. Another formulation of the same problem has been proposed in [11]. In both cases, the interaction matrix or image Jacobian (which links the motion of the features to camera motion) is proposed only for point features. Furthermore, the introduction of constraints in the camera trajectory is not considered within the proposed framework.

The introduction of constraints has received great attention in both the robotics (e.g., [4, 19]) and the computer-graphics [5] communities. The resulting solutions are often similar. Each constraint is defined mathematically as a function of the camera parameters (location and orientation) to be minimized using deterministic (e.g., gradient approaches) or stochastic (e.g., simulated annealing) optimization processes. These approaches feature numerous drawbacks. First they are usually time consuming (the search space is of dimension six) and the optimization has to be considered for each iteration of the animation process (i.e., for each new frame). It is then difficult to consider these techniques for reactive applications such as video games. As already stated, visual servoing allows the introduction of constraints in the camera trajectory [15, 16, 18]. These constraints are modeled as a cost function to be minimized. The resulting motion, also named the secondary task, is then projected in the null space of the main task; it has then no effect on the main visual task. As the camera trajectory that ensures both the task and the constraints is computed locally in this framework, it can be handled in real-time as required by the considered applications.

1.3 Presented system and contributions

The aim was to define the basic camera trajectories for virtual movie directors as well as the automatic control of a camera for reactive applications such as video games. We assume that we know the model of the scene fully at the current instant. Within this context, we present a complete framework, based on visual servoing, that allows the definition of positioning tasks with respect to a set of "virtual visual features" located within the environment (these features can be points, lines, spheres, cylinders, etc.). When the specified task does not constrain all the camera d.o.f., the method allows the introduction of secondary tasks that can be achieved under the constraint that the visual task is itself achieved. Furthermore the considered features are not necessarily motionless. Using this approach we present solutions to various non-trivial problems in computer animation. Some of these tasks are more concerned with reactive applications (target tracking and following, obstacles and occlusion avoidance), while others deal with cinema applications (panning, camera traveling, lighting-conditions optimization, etc.).

The remainder of this paper is organized as follows: Section 2 recalls the visual servoing framework within the task function approach. Section 3 presents methods allowing navigation in cluttered dynamic environments. Section 4 handles constraints more closely related to the cinema industry.

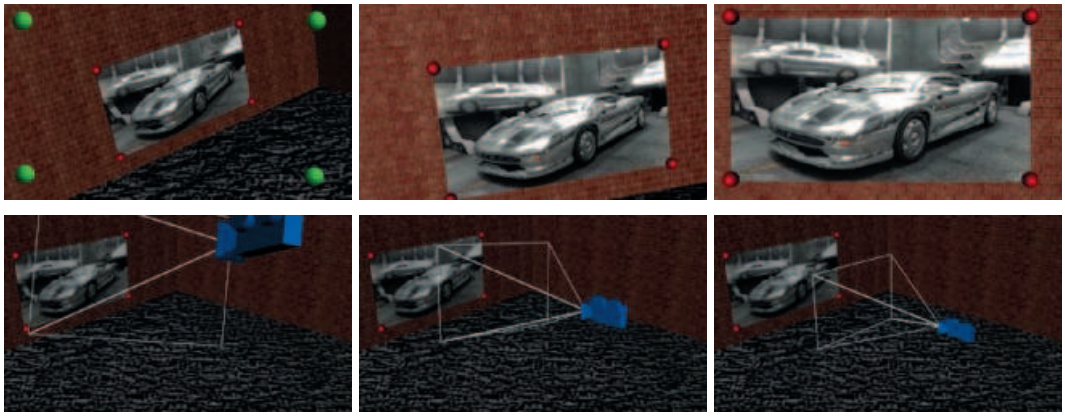


Fig. 1. Elementary positioning task: initial, intermediate and final frames acquired by the camera during the visual servo-loop. The *first line* shows the camera view and the *second line* shows an external view of the scene with the position of the camera. The *red points* are the visual features considered in the servo loop. The visual servoing loop aimed at minimizing the error between the current positions of these points (in *red*) and their desired positions (which appear in *green* in the initial frame)

2 Image-based camera control

Image-based visual servoing consists of specifying a task as the regulation in the image of a set of visual features [6, 8]. The basic idea of visual servoing is then to construct a servo loop that causes the camera to iteratively move toward its final position based on the contents of the image at each frame and the desired contents of the image. Let us insist on the fact that the specification of the camera's final position is not defined in the 3D space but is fully dependent on the 2D visual constraints defined by the visual servoing task.

Figure 1 depicts a concrete example of visual servoing servo-loop behavior. In this experiment, the goal is to see the photo of the car at a given location in the image. The red points are the visual features considered in the visual task. The visual servoing loop aimed at minimizing the error between the current positions of these points (in red) and their desired positions (which appear in green in the upper left image).

Embedding visual servoing in the task function approach [18] allows general results helpful in the analysis and synthesis of efficient closed-loop control schemes to be used. A good review and introduction to visual servoing can be found in [10].

2.1 Camera positioning with respect to visual targets

Let us denote $\boldsymbol{P}$ as the set of selected visual features used in the visual servoing task measured from the image, or by projection in the computer-graphics context, at each iteration of the control law. To ensure the convergence of $\boldsymbol{P}$ with its desired value $\boldsymbol{P}_{\mathrm{d}}$, we need to know the interaction matrix $\boldsymbol{L}_{\boldsymbol{P}}^{T}$, which links the motion of the object in the image to the camera motion. It is defined by the now classic equation [6]:

$$\dot{\boldsymbol{P}} = \boldsymbol{L}_{\boldsymbol{P}}^{T}(\boldsymbol{P}, \boldsymbol{p})\boldsymbol{T}_{\mathrm{c}}, \tag{1}$$

where $\dot{\boldsymbol{P}}$ is the time variation of $\boldsymbol{P}$ (the motion of $\boldsymbol{P}$ in the image) due to the camera motion $\boldsymbol{T}_{\mathrm{c}}$. The parameters $\boldsymbol{p}$ involved in $\boldsymbol{L}_{\boldsymbol{P}}^{T}$ represent the depth information between the considered objects and the camera frame. A vision-based task $\boldsymbol{e}_1$ is defined by:

$$\boldsymbol{e}_1 = \boldsymbol{C}(\boldsymbol{P} - \boldsymbol{P}_{\mathrm{d}}), \tag{2}$$

where $\boldsymbol{C}$, called the combination matrix, has to be chosen such that $\boldsymbol{C}\boldsymbol{L}_{\boldsymbol{P}}^{T}$ is full rank along the desired trajectory $r \in SE^3$. If $\boldsymbol{e}_1$ constrains the 6 d.o.f., it can be defined as $\boldsymbol{C} = \boldsymbol{L}_{\boldsymbol{P}}^{T^{+}}(\boldsymbol{P}, \boldsymbol{p})$. $\boldsymbol{L}^{+}$ is the pseudo inverse of matrix $\boldsymbol{L}$ and is defined by

- $\boldsymbol{L}^{+} = (\boldsymbol{L}^{T}\boldsymbol{L})^{-1}\boldsymbol{L}^{T}$ if the n d.o.f. are constrained (i.e.,, if rank $\boldsymbol{L} \geq 6$).
- $\boldsymbol{L}^{+} = \boldsymbol{L}^{T}(\boldsymbol{L}\boldsymbol{L}^{T})^{-1}$ if rank $\boldsymbol{L} \leq 6$.

We will see in Sect. 2.3 how to define $\boldsymbol{C}$ if the 6 d.o.f. are not constrained.
To make $\boldsymbol{e}_1$ decrease exponentially, the camera velocity given as input to the virtual camera is given by

$$\boldsymbol{T}_{\mathrm{c}} = -\lambda \boldsymbol{e}_1, \tag{3}$$

where λ is a proportional coefficient.
Within this framework we can easily perform positioning tasks with respect to any object in the scene. The main advantage of this approach is that, even if the task is specified within the 2D image space, the control is performed in 3D.

2.2 Building multiple image-based constraints

One of the difficulties in image-based visual servoing is to derive the interaction matrix $\boldsymbol{L}^T$ which corresponds to the selected control features. A systematic method has been proposed to analytically derive the interaction matrix from a set of control features defined based upon geometrical primitives [6]. Any kind of visual information can be considered within the same visual servoing task (coordinates of points, line orientation, surface or more generally inertial moments, distance, etc.).
Knowing these interaction matrices, the construction of elementary visual servoing tasks is straightforward. A large library of elementary skills can be proposed. The current version of our system allows one to define X-to-Y feature-based tasks (or constraint), where X and Y are defined in {point, line, sphere, cylinder, circle, etc.}. Defining X-to-Y feature-based tasks means that the operator wants to the object X "on" the object Y in the image space. Let us note that X and Y are not necessarily the same features, for example, a point-to-line constraint means that we want to see a 2D point on a 2D line.
From these elementary positioning constraints, more complex tasks can be considered by "stacking" the interaction matrix and the error vector related to each elementary task. For example, if we want to build a positioning task with respect to a segment, defined by two points $\boldsymbol{P}_1$ and $\boldsymbol{P}_2$, we define two point-to-point constraints. We thus want to minimize the error between the current position of $\boldsymbol{P}_1$ ($\boldsymbol{P}_2$) in the image and its desired position, $\boldsymbol{P}_{\mathrm{d}_1}$ ($\boldsymbol{P}_{\mathrm{d}_2}$).
The resulting interaction matrix as considered in 1 will be defined by "stacking" interaction matrices $\boldsymbol{L}_{\boldsymbol{P}_1}^T$ related to the first constraint and $\boldsymbol{L}_{\boldsymbol{P}_2}^T$ related to the second constraint. This leads to the new matrix $\boldsymbol{L}_{\boldsymbol{P}}^T$ defined by

$$\boldsymbol{L}_{\boldsymbol{P}}^T = \begin{bmatrix} \boldsymbol{L}_{\boldsymbol{P}_1}^T \\ \boldsymbol{L}_{\boldsymbol{P}_2}^T \end{bmatrix}, \tag{4}$$

where $\boldsymbol{L}_{\boldsymbol{P}_i}^T$ is defined, if $\boldsymbol{P}_i = (X, Y)$ and z is its depth, by (see [6] for its derivation):

$$\boldsymbol{L}_{\boldsymbol{P}}^T = \begin{pmatrix} -1/z & 0 & X/z & XY & -(1+X^2) & Y \\ 0 & -1/z & Y/z & 1+Y^2 & -XY & -X \end{pmatrix} \tag{5}$$

and the error vector involved in 2 is given by

$$(\boldsymbol{P} - \boldsymbol{P}_{\mathrm{d}}) = \begin{pmatrix} X_1 - X_{\mathrm{d}_1} \\ Y_1 - Y_{\mathrm{d}_1} \\ X_2 - X_{\mathrm{d}_2} \\ Y_2 - Y_{\mathrm{d}_2} \end{pmatrix}. \tag{6}$$

The full control law is then given by

$$\begin{aligned} \boldsymbol{T}_{\mathrm{c}} = -\lambda & \\ \times & \begin{pmatrix} -1/z_1 & 0 & X_1/z_1 & X_1Y_1 & -(1+X_1^2) & Y_1 \\ 0 & -1/z_1 & Y_1/z_1 & 1+Y_1^2 & -X_1Y_1 & -X_1 \\ -1/z_2 & 0 & X_2/z_2 & X_2Y_2 & -(1+X_2^2) & Y_2 \\ 0 & -1/z_2 & Y_2/z_2 & 1+Y_2^2 & -X_2Y_2 & -X_2 \end{pmatrix}^+ \\ & \times \begin{pmatrix} X_1 - X_{\mathrm{d}_1} \\ Y_1 - Y_{\mathrm{d}_1} \\ X_2 - X_{\mathrm{d}_2} \\ Y_2 - Y_{\mathrm{d}_2} \end{pmatrix}. \end{aligned} \tag{7}$$

More positioning constraints can thus be simply defined considering the appropriate interaction matrix and error vector.

2.3 Introducing constraints within the positioning task

If the vision-based task does not constrain all the n robot d.o.f., a secondary task (which usually represents a camera-trajectory constraint) can be performed. $\boldsymbol{C}$ is now defined as $\boldsymbol{C} = \boldsymbol{W}\boldsymbol{L}_{\boldsymbol{P}}^{T^+}$ and we obtain the following task function:

$$\boldsymbol{e} = \boldsymbol{W}^+\boldsymbol{e}_1 + (\boldsymbol{I}_n - \boldsymbol{W}^+\boldsymbol{W})\boldsymbol{e}_2, \tag{8}$$

where

- $\boldsymbol{e}_2$ is a secondary task. Usually $\boldsymbol{e}_2$ is defined as the gradient of a cost function h_{s} to be minimized ($\boldsymbol{e}_2 = \frac{\partial h_{\mathrm{s}}}{\partial \boldsymbol{r}}$). This cost function is minimized under the constraint that $\boldsymbol{e}_1$ is realized.

- $\boldsymbol{W}^+$ and $\boldsymbol{I}_n - \boldsymbol{W}^+\boldsymbol{W}$ are two projection operators which guarantee that the camera motion due to the secondary task is compatible with the regulation of $\boldsymbol{P}$ to $\boldsymbol{P}_\mathrm{d}$. $\boldsymbol{W}$ is a full rank matrix such that $\mathrm{Ker}\boldsymbol{W} = \mathrm{Ker}\boldsymbol{L}_{\boldsymbol{P}}^T$. Thanks to the choice of matrix $\boldsymbol{W}$, $\boldsymbol{I}_n - \boldsymbol{W}^+\boldsymbol{W}$ belongs to $\mathrm{Ker}\boldsymbol{L}_{\boldsymbol{P}}$, which means that the realization of the secondary task will have no effect on the vision-based task ($\boldsymbol{L}_{\boldsymbol{P}}^T(\boldsymbol{I}_n - \boldsymbol{W}^+\boldsymbol{W})\boldsymbol{e}_2 = 0$). Let us note that if the visual task constrains all the n d.o.f. of the manipulator, we have $\boldsymbol{W} = \boldsymbol{I}_n$, which leads to $\boldsymbol{I}_n - \boldsymbol{W}^+\boldsymbol{W} = 0$. It is thus impossible in that case to consider any secondary task.

The control is now given by:

$$\boldsymbol{T}_\mathrm{c} = -\lambda\boldsymbol{e} - (\boldsymbol{I}_n - \boldsymbol{W}^+\boldsymbol{W})\frac{\partial \boldsymbol{e}_2}{\partial t}. \tag{9}$$

2.4 Tracking a mobile target

A target motion generally induces tracking errors that have to be suppressed in order to always achieve the tracking task perfectly.
In that case, the motion of the target in the image can be rewritten as

$$\dot{\boldsymbol{P}} = \boldsymbol{L}_{\boldsymbol{P}}^T\boldsymbol{T}_\mathrm{c} - \boldsymbol{L}_{\boldsymbol{P}}^T\boldsymbol{T}_0, \tag{10}$$

where $\boldsymbol{L}_{\boldsymbol{P}}^T\boldsymbol{T}_\mathrm{c}$ and $\boldsymbol{L}_{\boldsymbol{P}}^T\boldsymbol{T}_0$ are respectively the contributions of the camera velocity and of the autonomous target motion to the motion of the target in the image. The new camera velocity that suppresses the tracking errors is then given by

$$\boldsymbol{T}_\mathrm{c} = -\lambda\boldsymbol{e} - (\boldsymbol{I}_n - \boldsymbol{W}^+\boldsymbol{W})\frac{\partial \boldsymbol{e}_2}{\partial t} - \alpha\boldsymbol{T}_0, \tag{11}$$

where $\alpha \in [0, 1]$ is a scalar. If $\alpha = 1$ the tracking errors are fully suppressed, while if $\alpha = 0$ they are not handled.

3 Reactive viewpoint planning

The positioning tasks that can be considered within the framework presented in the previous section are quite simple. As we did not consider the environment, the target was assumed to be "alone". We now present a method that makes it possible to achieve far more complex tasks in dynamic *cluttered environments*. In this difficult context we will propose a purely reactive framework in order to avoid undesirable configurations in an animation context.

3.1 Avoiding obstacles

Obstacle avoidance is a good example of what can be easily specified within the proposed framework. Let us assume that the camera is moving in a cluttered environment while its view is centered on a visual target. The goal is to ensure this task while avoiding all the obstacles in the scene.
There are in fact multiple solutions to this problem: one solution is to plan a trajectory that avoids the obstacles using a trajectory planning process. Another solution is to consider a secondary task that uses the redundant d.o.f. of the camera to move away from obstacles. This function will tend to maximize the distance between the camera and the obstacle. A good cost function to achieve the goal should be maximum (infinite) when the distance between the camera and the obstacle is null. The simplest cost function is then given by

$$h_\mathrm{s} = \alpha\frac{1}{2\|C - O_\mathrm{c}\|^2} \tag{12}$$

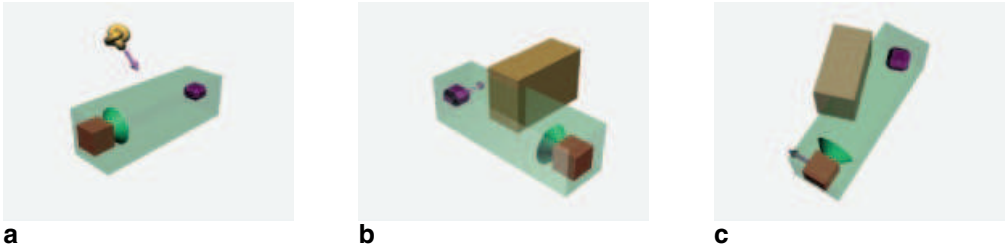
where $C(0, 0, 0)$ is the camera location and $O_\mathrm{c}(x_\mathrm{c}, y_\mathrm{c}, z_\mathrm{c})$ are the coordinates of the closest obstacle to the camera, both expressed in the camera frame. (Note that any other cost function that reflects a similar behavior suits the problem.) If $O_\mathrm{s}(x_\mathrm{s}, y_\mathrm{s}, z_\mathrm{s})$ are the coordinates of the obstacle within the scene frame (or reference frame) and $\boldsymbol{M}_\mathrm{c}(\boldsymbol{RT})$ is the homogeneous matrix that describes the camera position within this reference frame, the obstacle coordinates within the camera frame are given by $\boldsymbol{X}_\mathrm{c} = \boldsymbol{R}^T\boldsymbol{X}_\mathrm{s} - \boldsymbol{R}^T\boldsymbol{T}$.
The components of the secondary task are given by

$$\boldsymbol{e}_2 = -(x_\mathrm{c}, y_\mathrm{c}, x_\mathrm{c}, 0, 0, 0)^T\frac{h_\mathrm{s}^2}{\alpha} \quad \text{and} \quad \frac{\partial \boldsymbol{e}_2}{\partial t} = 0. \tag{13}$$

Multiple obstacles can be handled considering the cost function $h_\mathrm{s} = \sum_i \alpha\frac{1}{\|C - O_{c_i}\|^2}$.

3.2 Avoiding occlusions

The goal here is to avoid the occlusion of the target due to static or moving objects (with unknown motion). The virtual camera has to perform adequate motion in order to avoid the risk of occlusion while taking into account the desired constraints between the camera and the target. Related work is described in [13]. There are actually many situations that may evolve into occlusion. The first and most simple case is a moving object that crosses the



a **b** **c**

Fig. 2a,b. Occlusion issues. **a** Occlusion due to a moving object; **b** occlusion due to the target motion; and **c** occlusion due to the camera motion

camera–target line (see Fig. 2a). Two other similar cases may be encountered: in the first one (Fig. 2b) the target moves behind another object in the scene, while in the second one (Fig. 2c) the camera follows an undesirable trajectory and is hidden behind an object.

We will now first present a general image-based approach that makes it possible to generate adequate camera motion automatically to avoid occlusions [15]. Second, we will see a simple method to determine the risk of occlusion in order to weight adequately the camera response (i.e., its velocity).

Automatic generation of adequate motions

Let us consider the set $\mathcal{O}$ of objects in the scene which may occlude the target T: $\mathcal{O} = \{O_1, \ldots O_n\}$. According to the methodology presented in Sect. 2.3, we have to define a function h_s which reaches its maximum value when the target is occluded by another object of the scene. In fact this occlusion problem can be fully defined in the image. If the occluding object is closer than the target when the distance between the projection of the target and the projection of the occluding object decreases, the risk of occlusion increases.

A function similar to the one proposed in (12) may be considered. In this section, in order to avoid velocities that are too important, we define h_s as an exponential of the distance in the image:

$$h_s = \frac{1}{2}\alpha \sum_{i=1}^{n} \mathrm{e}^{-\beta\left(\|\mathrm{pr}(T) - pr(O_i)\|^2\right)}, \tag{14}$$

where pr(.) is the projection function and α and β are two scalar constants. α sets the amplitude of the control law due to the secondary task. The components of $\boldsymbol{e}_2$ and $\frac{\partial \boldsymbol{e}_2}{\partial t}$ involved in (9) are then

$$\boldsymbol{e}_2 = \frac{\partial h_s}{\partial \boldsymbol{r}} = \frac{\partial h_s}{\partial \boldsymbol{P}} \frac{\partial \boldsymbol{P}}{\partial \boldsymbol{r}}, \qquad \frac{\partial \boldsymbol{e}_2}{\partial t} = 0.$$

Computing $\frac{\partial h_s}{\partial \boldsymbol{P}}$ is seldom difficult. $\frac{\partial \boldsymbol{P}}{\partial \boldsymbol{r}}$ is nothing but the interaction matrix $L_{\boldsymbol{P}}^T$.

Let us consider the case of a single occluding object, here considered as a point. The generalization to other types of object and/or to multiple objects is straightforward. We want to see the target T at a given location in the image. Thus we will consider the coordinates $\mathrm{pr}(T) = \boldsymbol{P} = (X, Y)$ as the projection of its center of gravity. Treatment of O may be slightly different; we will not consider the projection of its center of gravity but the 2D point $\boldsymbol{P}_O = (X_O, Y_O)$ of its projection in the image plane the closest to pr(T). The cost function is then given by

$$h_s = \frac{1}{2}\alpha \mathrm{e}^{-\beta\|\boldsymbol{P} - \boldsymbol{P}_O\|^2},$$

and $\boldsymbol{e}_2$ is given by

$$\boldsymbol{e}_2 = \frac{\partial h_s}{\partial \boldsymbol{r}} = \frac{\partial h_s}{\partial X}\boldsymbol{L}_X^T + \frac{\partial h_s}{\partial Y}\boldsymbol{L}_Y^T, \tag{15}$$

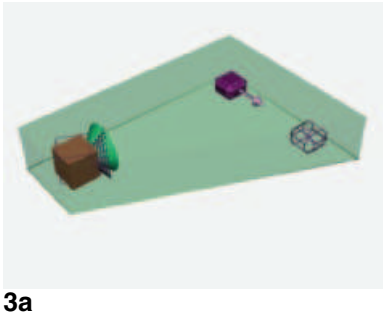
where

$$\frac{\partial h_s}{\partial X} = -\alpha\beta(X - X_O)\mathrm{e}^{-\beta\|\boldsymbol{P} - \boldsymbol{P}_O\|^2}$$

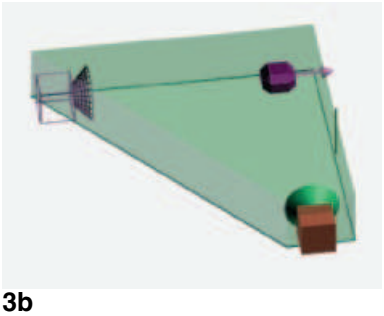
and

$$\frac{\partial h_s}{\partial Y} = -\alpha\beta(Y - Y_O)\mathrm{e}^{-\beta\|\boldsymbol{P} - \boldsymbol{P}_O\|^2}.$$

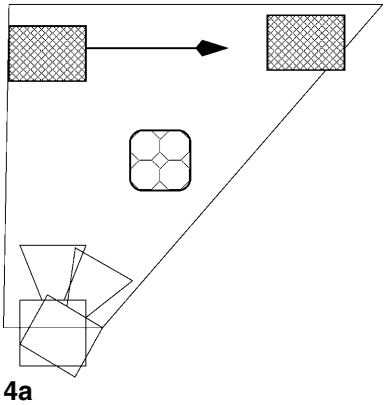
In fact $\boldsymbol{e}_2$ as defined in 15 is an approximation of $\frac{\partial h_s}{\partial \boldsymbol{r}}$. Indeed $\boldsymbol{L}_{\boldsymbol{P}}^T = \left[\boldsymbol{L}_X^T, \boldsymbol{L}_Y^T\right]^T$ is the interaction matrix related to a physical point. In our case, since the point is defined as the closest projected point of O to T, the corresponding physical point in the 3D space will change over time. However considering $\boldsymbol{L}_X^T$ and $\boldsymbol{L}_Y^T$ in 15 is locally a good approximation.



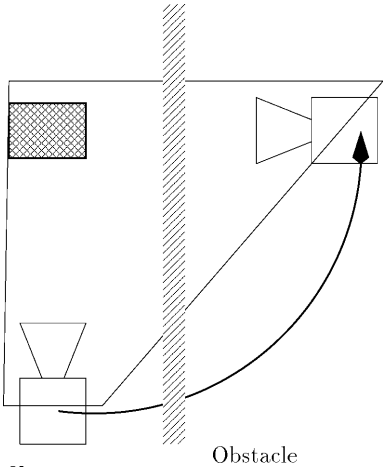
3a



3b



4a



4b

Fig. 3a,b. Computing the risk of occlusion
Fig. 4a,b. Detection of a future (**a**) occlusion (**b**) collision with an obstacle

Risk of occlusion

Using the presented approach to compute the camera reaction is fine if the occluding object moves between the camera and the target [15] as depicted in Fig. 2. Indeed, in that case occlusion will occur if no action is taken. However, it is neither necessary nor desirable to move the camera in all the cases (if the occluding object is farther than the target). A key point is therefore to detect if an occlusion may actually occur. In that case we first compute a bounding volume $\mathcal{V}$ that includes both the camera and the target at the current time t and its predicted position at time $t+n\mathrm{d}t$ assuming a constant target velocity (see Figs. 3 and 4). $\mathrm{d}t$ is the time interval between two intersection tests. An occlusion will occur if an object is located within this bounding box. The time-to-occlusion may be computed as the largest n for which the bounding box is empty. Precision for the computation of the largest n depends on $\mathrm{d}t$: the smaller $\mathrm{d}t$ is, the greater the estimation of the time-to-intersection will be. If an object $\mathcal{O}$ of the scene is in motion, in the same way, we consider the intersection of the volume $\mathcal{V}$ with a bounding volume that includes O at time t and at time $t+n\mathrm{d}t$.

Let us point out two other interesting issues:

- Obstacle avoidance may be considered in this context. Indeed, if an obstacle is on the camera trajectory, it will be located in the created bounding box (see Fig. 4b). The system will therefore forbid the camera to move in that direction.
- Some cases are more difficult to handle. A good example is a target moving in a corridor (see Fig. 5). In that case, the only solution to avoid the occlusion of the target by one of the walls and to avoid contact with the other wall is to reduce the camera–target distance. This can only be done if the z-axis is not controlled by the primary task.

In conclusion, let us note that in this subsection, we have proposed a method to detect and quantify the

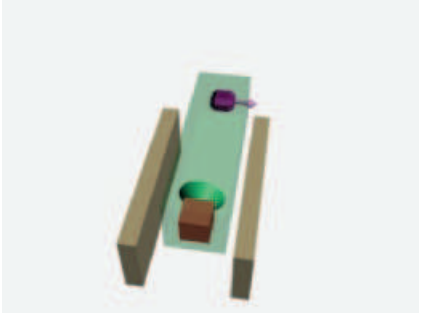


Fig. 5. Occlusion issues: camera in a corridor

risk of occlusion. The method proposed in the previous subsection must be, in all cases, used to generate the adequate motion that will actually avoid occlusion. The time-to-occlusion computed here will in fact be used to set the parameter α [see (14)], which tunes the amplitude of the response to the risk.

4 Virtual director for automatic cinematography

Whereas the issues considered in the previous section are more related to reactive applications such as video games, the problems considered in this section are more concerned with camera control for movie-making applications. The question considered here is the following: Where should we place the camera to ensure film constraints within a given *shot* [1]? Our goal here is not to provide a director with a language that describes scenes and shots such as in [3], but to propose some elementary skills to be used afterward by the director [9, 17].

4.1 Cinematographic basic camera placement

Panning and tracking

Panning and tracking, certainly the most common camera motions, are straightforward to consider within the image-based framework and have been widely considered in the previous sections of this paper. In fact the only difficulty is to choose the visual features (virtual or not) on which we want to servo. This choice is very important as it will determine the d.o.f. of the virtual camera that will be used to achieve the task. For example for panning issues, the users are likely to choose one or two virtual points or a straight line as visual features (for these features the pan axes of the camera will be controlled). For tracking issues, the adequate features may depend on the desired camera motion. For example, if the camera motion has to be "parallel" to the target trajectory, the 6 d.o.f. must be constrained in order to achieve a rigid link between the camera and the target (4 points or 4 lines – or any other combination of visual features such that $\boldsymbol{L}^T$ is a full rank-6 matrix – are then suitable for such a purpose).

Trajectory tracking

As regards the trajectory tracking issue, the problem is fairly simple. We want the camera to move on a curve $\mathcal{V}(t) = (x(t), y(t), z(t))$ defined in the camera frame. We consider a secondary task that is nothing but a function of the distance between the camera and the point $\mathcal{V}(t)$. A good solution is to define the secondary task as the function h_s simply defined as

$$h_s = \|\mathcal{V}(t)\|^2. \tag{16}$$

Many other basic cinematographic issues exist (see [1, 9, 17]), e.g., building apex camera placement (a situation considering two actors that places the camera so that the first actor is centered on one side of the screen and the second is centered on the other; this task can be defined by two segments or two points), external or internal view (which has to consider the target and a virtual line of interest). Our goal is not to describe these tasks here. However, as they are described within the image space, image-based camera control is suitable for such issues.

4.2 Controlling lighting conditions

Controlling lighting conditions (i.e., the "photography" problem) is a fundamental and nontrivial issue for a film director. The main problem is to define what a good shot is with respect to these conditions. Two different goodness functions may be proposed to achieve this goal: one is directly based on the intensity within the image, while the second is based on the intensity gradient. The former is obviously not a useful goal from a cinematographer's point of view. Indeed, usually this will result in a light source at or close to the camera position and may lead to a position with many undesirable qualities (such as specularities). The latter is more useful, since it con-

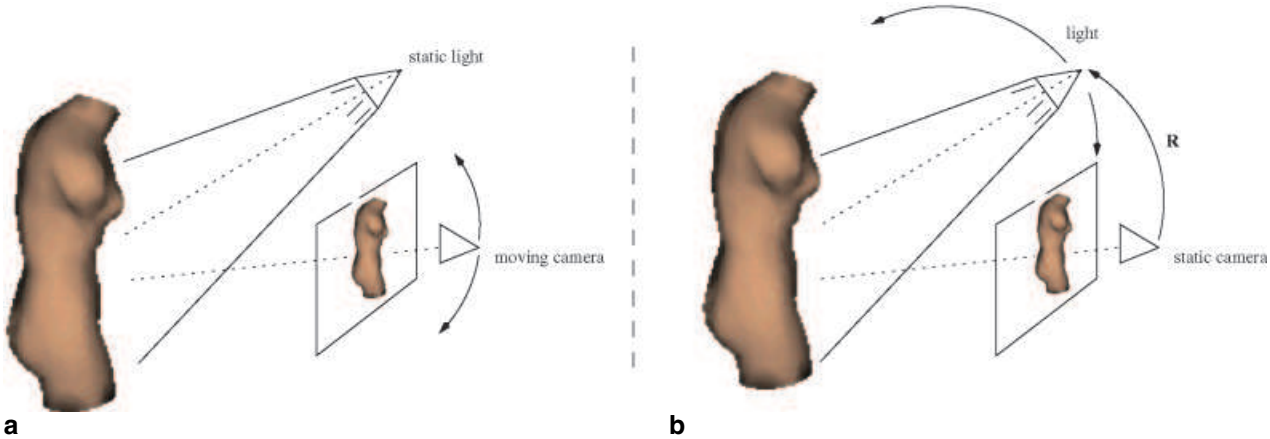


Fig. 6a,b. Controlling lighting conditions. **a** static light/moving camera **b** moving light/static camera

siders information about the contrast of the resulting image.

To outline the issue, our primary goal will be to move the camera while the light remains static (see Fig. 6a). Then, we will propose to move the light while the camera remains static (see Fig. 6b).

Modeling

Though it is not the most interesting situation, our first goal is to position the camera with respect to the lit aspect of the object. Therefore, we want to maximize the quantity of light (re-)emitted by this object to ensure good lighting conditions. Applying the methodology proposed in the previous sections, we want to maximize the following cost function:

$$h_s = \frac{1}{n} \sum_X \sum_Y I(X, Y),$$

where $I(X, Y)$ represents the intensity of the 2D point (X, Y) and n is the number of points (X, Y) that belong to the object. The secondary task is then given by

$$\begin{aligned} \frac{\partial h_s}{\partial \boldsymbol{r}} &= \frac{1}{n} \sum_X \sum_Y \left(\frac{\partial h_s}{\partial X} \frac{\partial X}{\partial \boldsymbol{r}} + \frac{\partial h_s}{\partial Y} \frac{\partial Y}{\partial \boldsymbol{r}} \right) \\ &= \frac{1}{n} \sum_X \sum_Y \left(\nabla_X \boldsymbol{L}_X^T + \nabla_Y \boldsymbol{L}_Y^T \right), \end{aligned} \tag{17}$$

where $\nabla I_X = \frac{\partial I}{\partial X}$ and $\nabla I_Y = \frac{\partial I}{\partial Y}$ represent the spatial intensity gradient.

A more interesting goal is to maximize the contrast within the image; one possible criterion will be to maximize the sum of the spatial intensity gradient within the image. The corresponding cost function is given by

$$h_s = \frac{1}{n} \sum_X \sum_Y \left[\nabla I_X^2 + \nabla I_Y^2 \right]. \tag{18}$$

We therefore need to compute the gradient $\frac{\mathrm{d}h_s}{\mathrm{d}r}$, which is in fact given by

$$\frac{\mathrm{d}h_s}{\mathrm{d}r} = \frac{1}{n} \sum_X \sum_Y \left(\frac{\mathrm{d}h_s}{\mathrm{d}X} \boldsymbol{L}_X^T + \frac{\mathrm{d}h_s}{\mathrm{d}Y} \boldsymbol{L}_Y^T \right). \tag{19}$$

After some rewriting, we finally obtain

$$\begin{aligned} \frac{\mathrm{d}h_s}{\mathrm{d}r} = \frac{2}{n} \sum_X \sum_Y \Bigg[&\left(\frac{\mathrm{d}^2 I}{\mathrm{d}X^2} \nabla I_X + \frac{\mathrm{d}^2 I}{\mathrm{d}Y\,\mathrm{d}X} \nabla I_Y \right) \boldsymbol{L}_X^T \\ &+ \left(\frac{\mathrm{d}^2 I}{\mathrm{d}X\,\mathrm{d}Y} \nabla I_X + \frac{\mathrm{d}^2 I}{\mathrm{d}Y^2} \nabla I_Y \right) \boldsymbol{L}_Y^T \Bigg]^T. \end{aligned} \tag{20}$$

Moving the light

Considering a static light and a mobile camera is not the most interesting context. Indeed if the camera is moving, the aspect of the object will change over time. It would be more interesting to control the light position and orientation while the camera remains static.

Here again we consider the visual servoing framework to point the light toward the object of interest and to achieve well-lit conditions. We first add

a virtual camera (with the same location and direction) to the light. The main task is specified as a simple focusing task that constrains the rotation d.o.f. of the virtual camera/light system. We then consider the redundancy to control the translation d.o.f. of the camera/light to impose the correct illumination of the object within the image acquired by the other camera. The task function is then defined as

$$\boldsymbol{e} = \boldsymbol{W}^{+}\boldsymbol{W} \underbrace{\boldsymbol{L}^{T^{+}}(\boldsymbol{P}-\boldsymbol{P}_{\mathrm{d}})}_{\text{main focusing task}} + (\boldsymbol{I}-\boldsymbol{W}^{+}\boldsymbol{W}) \underbrace{\begin{pmatrix} \boldsymbol{R} & -\boldsymbol{R}\,Skew(-\boldsymbol{R}^{T}\boldsymbol{T}) \\ 0 & \boldsymbol{R} \end{pmatrix} \frac{\mathrm{d}h_{\mathrm{s}}}{\mathrm{d}r}}_{\substack{\text{secondary task defined} \\ \text{with respect to the other camera}}}. \tag{21}$$

where $\boldsymbol{R}$ and $\boldsymbol{T}$ denotes the rotational and translational mappings of the camera frame onto the light frame.

Let us note here that if the camera is now moving, the problem remains exactly the same as long as we know the transformations $\boldsymbol{R}$ and $\boldsymbol{T}$ between the camera and the light.

5 Results

In this section some results are presented to illustrate our approach. Most of the images are generated in "real-time" (i.e., less than 0.1 s/frame without texture-mapping) on a simple SUN Ultra Sparc (170 Mhz) using Mesa GL (the images produced using this process can be seen in, for example, Fig. 13). The animations in Figs. 7, 10 and 11 integrate texture mapping and are not generated in real-time (however, the program was not optimized at all), while Fig. 8 was computed afterward using Maya from Alias Wavefront.

5.1 Elementary positioning tasks

We present in this paragraph experiments related to very simple positioning tasks: a positioning task with respect to four points and with respect to a segment with a trajectory tracking. Although such tasks are widely described in the robotics literature e.g., [6, 10] and in computer graphics papers [7, 11], we propose here some results for illustration issue.

Positioning with respect to a rectangle: rigid link

The goal of this experiment is to see a rectangle at a given position in the image. To achieve this task we have to control all the d.o.f. of the camera. One way (among others) to achieve this task is to define the rectangle as four points. This will give us the following interaction matrix:

$$\boldsymbol{L}_{\boldsymbol{P}}^{T} = \left[\boldsymbol{L}_{\boldsymbol{P}_1}^{T}, \quad \boldsymbol{L}_{\boldsymbol{P}_2}^{T}, \quad \boldsymbol{L}_{\boldsymbol{P}_3}^{T}, \quad \boldsymbol{L}_{\boldsymbol{P}_4}^{T}\right]^{T} \tag{22}$$

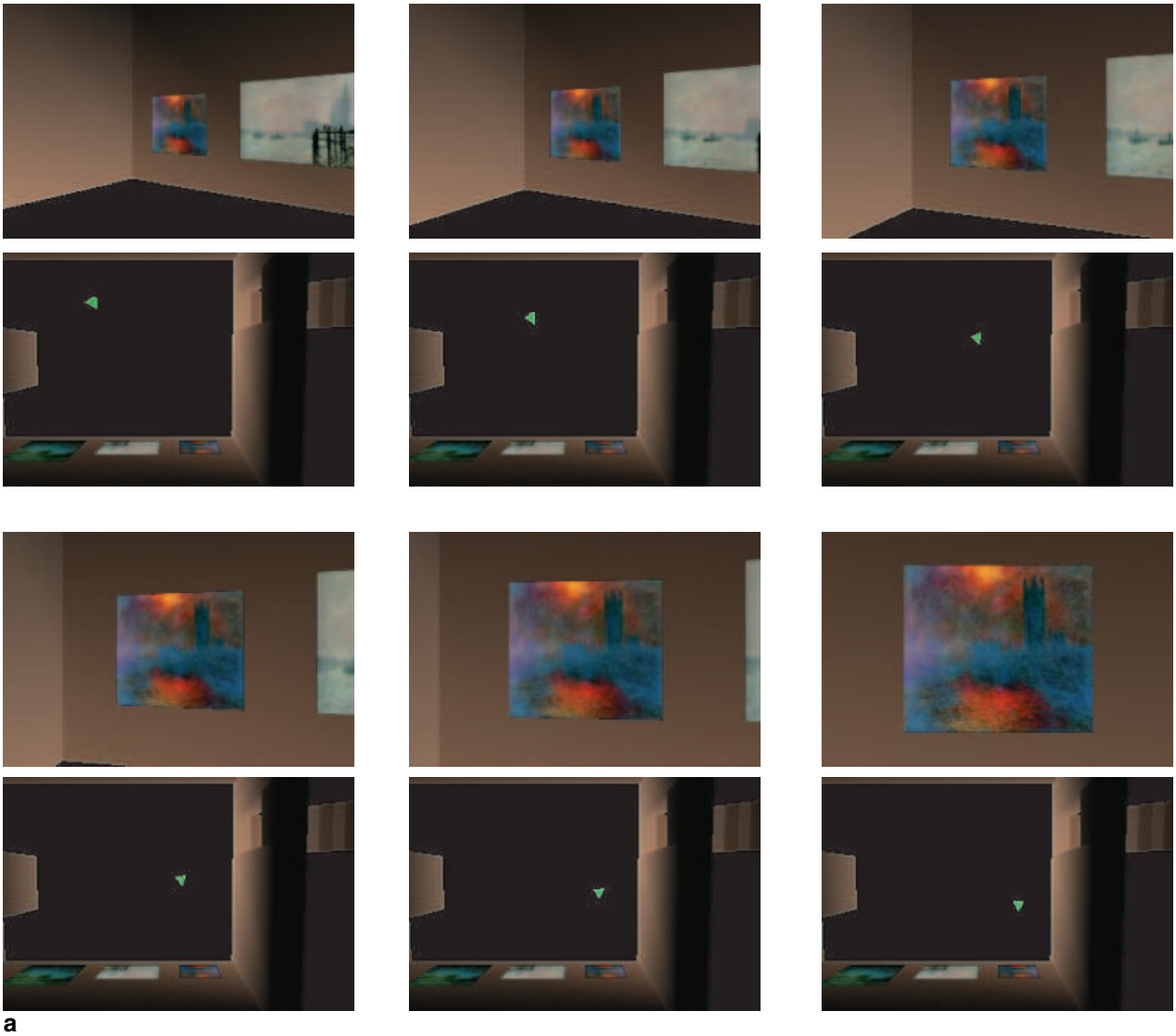
with $\boldsymbol{L}_{\boldsymbol{P}_i}^{T}, i = 1 \ldots 4$ defined as in (5). $\boldsymbol{L}_{\boldsymbol{P}}^{T}$ is then a 8×6 matrix and is full rank 6. Therefore the 6 d.o.f. are controlled.

In Fig. 7 we consider the painting by Monet as the object of interest. We want to see it centered in the screen. We therefore defined the desired position as four points defined by (a, b), $(-a, b)$, $(-a, -b)$, $(a, -b)$, where a and b are functions of the real painting size and of the desired painting size in the image. Figure 7a shows 6 key frames of a 600 frames animation of the camera view along with a bird's eye view that shows both the camera (in green) and the painting. Figure 7b and c show the camera velocities in translation and in rotation, while Fig. 7d depicts the error *in the image* between the current position of the painting and the desired one. Let us note that as the 6 d.o.f. are constrained by the camera, no additional constraint can be added on the camera trajectory.

Trajectory tracking

In the experiment described in Fig. 8, the camera focuses on the tower (i.e., we want to see this tower vertically and centered in the image). Let us note here that a similar task has been considered in [7].

Let us first consider the positioning task itself. It can be handled in various ways according to the chosen visual features. The simplest way to define a segment is to consider its two extremities. In that case $\boldsymbol{L}_{\boldsymbol{P}}^{T}$ is a full rank-4 matrix. The distance between the camera and the middle of the segment must remain constant. If we want to follow a trajectory that does not ensure this constraint, we will have to modify the focal length of the camera to ensure both the main task and the trajectory tracking [7]. This



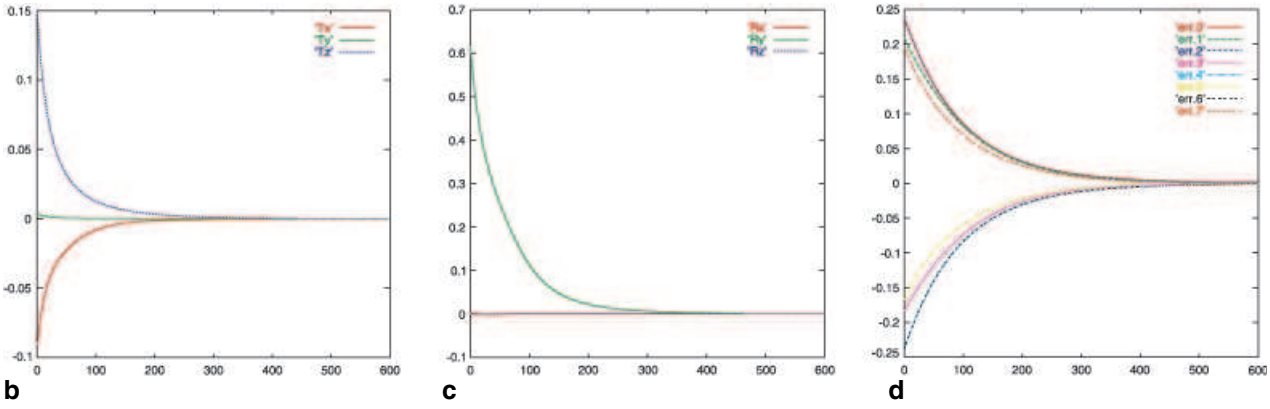


Fig. 7a–d. Positioning with respect to a painting (visual features are the four corners of the rectangle). **a** Six camera and bird's eye views; **b** camera translational velocities; **c** camera rotational velocities; **d** errors $\boldsymbol{P} - \boldsymbol{P}_{\mathrm{d}}$ in the image

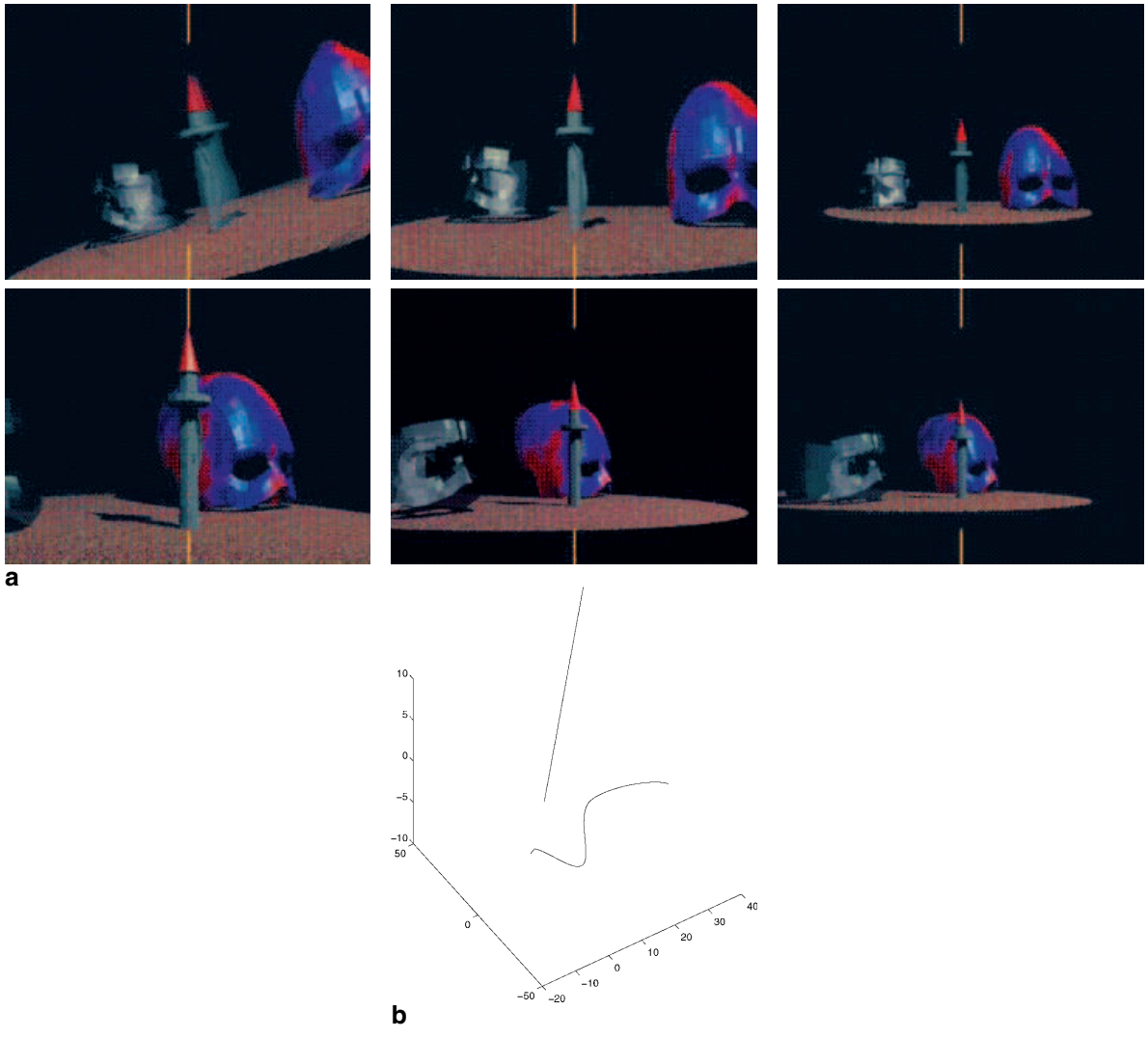


Fig. 8a,b. Positioning wrt. a line and trajectory tracking. **a** View of the camera; **b** 3D camera trajectory wrt. the line

solution is usually not suitable for cinematographic issues. The other way to consider this segment is to choose the segment support straight line as a visual feature. In that case, the interaction matrix is a full rank-2 matrix and only two d.o.f. are then constrained (note that a similar solution would have been to constrain the orientation of the segment and the position of its middle point). The first two frames of Fig. 8a show the beginning and the end of this focusing task. Once this is achieved, the camera follows a given 3D trajectory. The tracked trajectory is plotted on Fig. 8b.

5.2 Avoiding occlusions: museum walkthrough

In this example, we applied the proposed methodology to a navigation task in a complex environment. The target to be followed is moving in a museum-like environment. This "museum" has two rooms linked by stairs. The experiment goal is to keep the target in view (i.e., to avoid occlusions) while considering *online* the modifications of the environment (i.e., other moving objects).

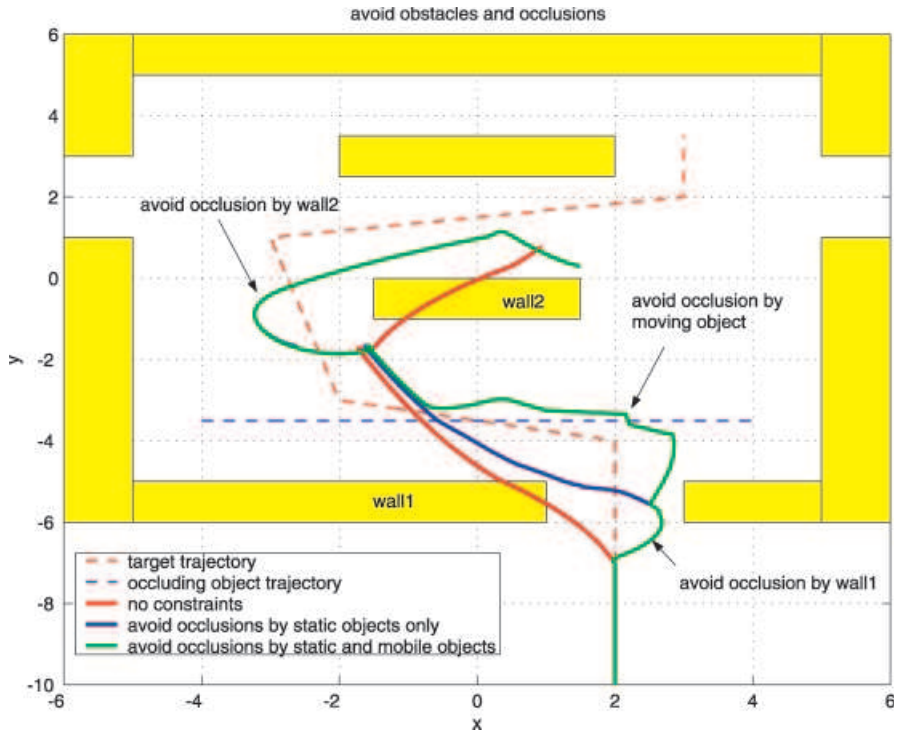


Fig. 9. Museum walk-through: camera trajectories for various strategies

We do not address in this paper the definition of the target trajectory. Finding a path for the target is a planning problem on its own. Solutions are proposed in, e.g., [5, 14]. Most of these approaches are based on a global path planning strategy (usually based on the potential field approach [12]).

In this example, we consider a focusing task with respect to an image-centered virtual sphere that has to be centered in the image. This task constrains 3 d.o.f. of the virtual camera (i.e., to achieve the focusing task and to maintain the radius constant in the image). The reader can refer to [6] for the complete derivation of the interaction matrix related to a sphere. Figure 9 shows the camera trajectories for various applied strategies while target and camera are moving in the first room of the environment. Obstacles appear in yellow. The target trajectory is represented as a red dotted line, while the trajectory of another moving object is represented as a blue dotted line. The red trajectory represents the simplest strategy: just focus on the object. As nothing is done to consider the environment, occlusions and then collisions with the environment occur. The blue trajectory only considers the avoidance of occlusions by static objects; as a consequence, occlusion by the moving object occurs. The green trajectory considers the avoidance of occlusions by both static and moving objects.

Figure 10 shows the views acquired by the camera if no specific strategy is considered to avoid occlusion of the target and obstacle avoidance. This leads to multiple occlusions of the target and multiple collisions with the environment. In Figs. 11 and 12 the control strategy considers the presence of obstacles. This time, the target always remains in the field of view and at its desired position in the image. The collisions with the wall and the occlusions of the target are correctly avoided. Let us note that the environment is not flat, and neither the target nor the camera move within a plane (the target "gets down" stairs in the last row of Fig. 11). The tracking and avoidance processes perform well despite the fact that the target moves in 3D. From the bird's eye view the yellow volume (associated with the camera–target couple) corresponds to the bounding volumes used to predict the occlusions.

Fig. 10. Museum walkthrough. The occlusions/obstacles avoidance process is not considered. This leads to multiple occlusions of the target and multiple collisions with the environment

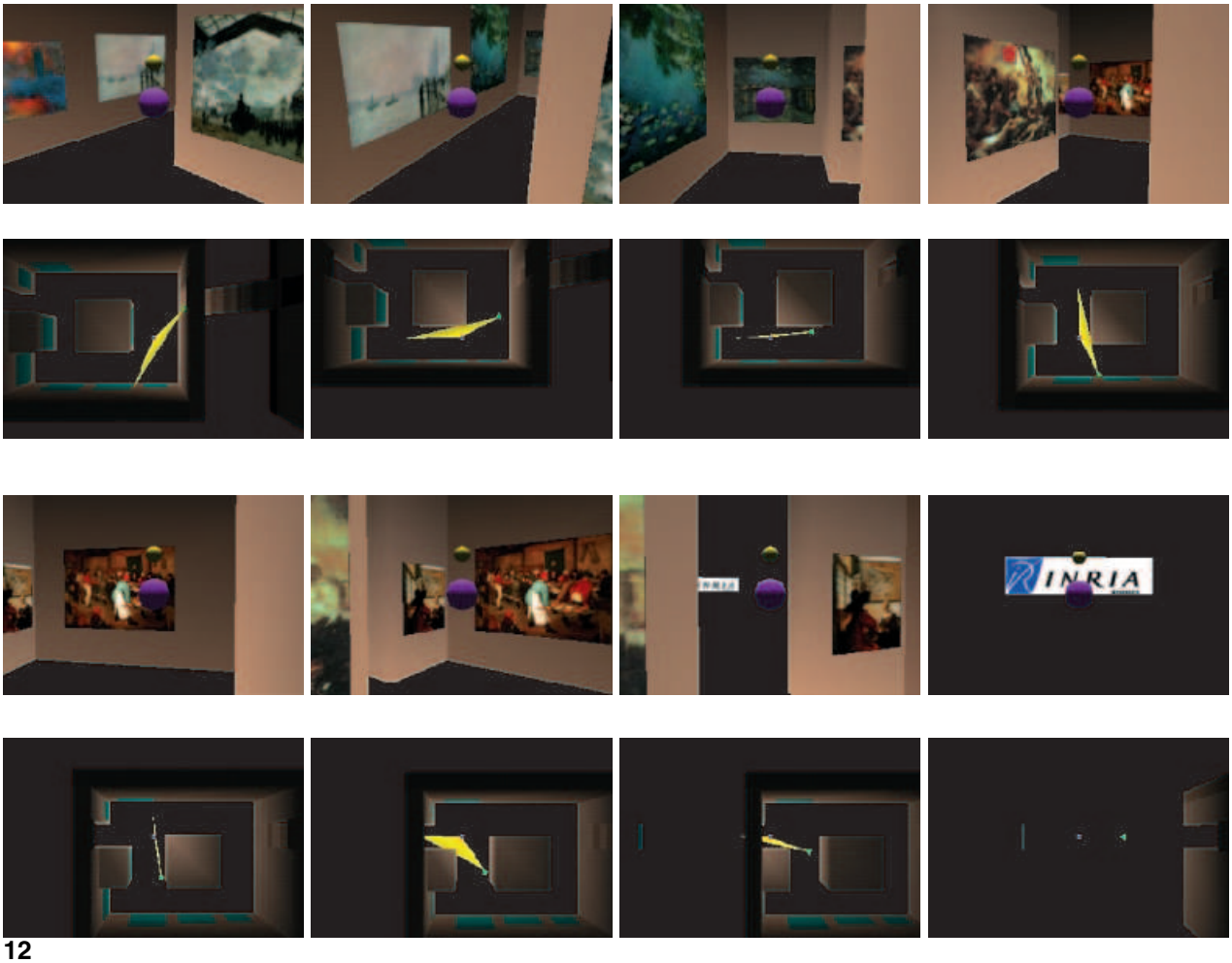
5.3 Walking in a corridor: Merging multiple constraints

In this experiment the considered task is the same but the target is moving within a narrow corridor and is turning right (see Fig. 13). It is not possible to achieve this task if the distance between the camera and the target remains constant. If one wants the camera to keep the target in view, an occlusion avoidance process has to be performed. The problem is that the motion computed to avoid the occlusion moves the camera toward the red wall. An obstacle avoidance process is then necessary. We then have three secondary tasks: one related to the camera–target distance, one related to obstacle avoidance (see Sect. 3.1) and the last one related to occlusion avoidance (see Sect. 3.2). The resulting control law automatically produces a motion that moves the camera away from the wall and reduces the camera–target distance. This distance, initially set to 3.5 m, decreases and reaches less that 2.5 m to ensure the task.

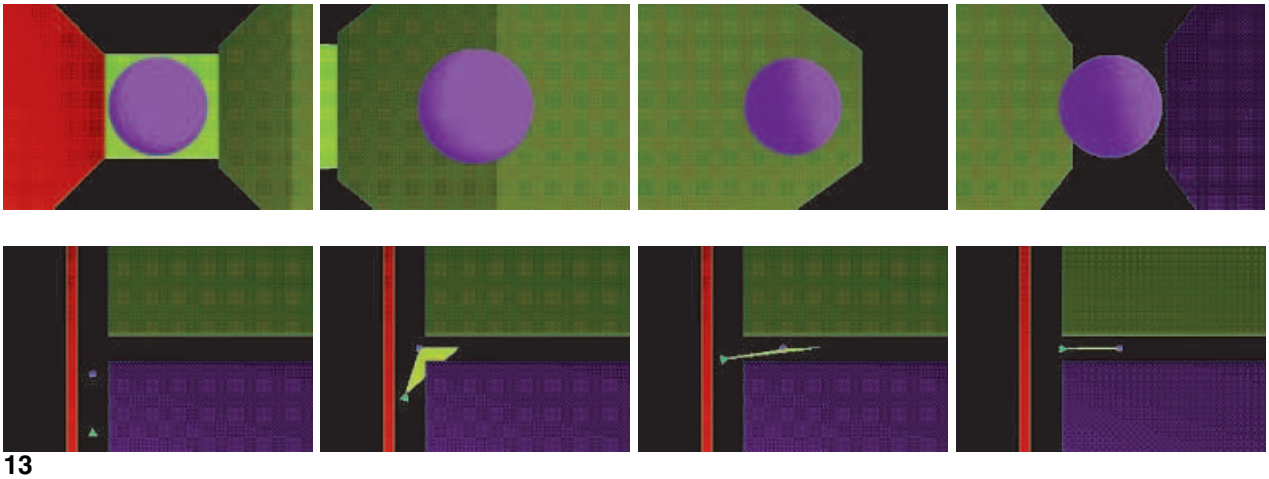
Fig. 11. Museum walkthrough (part 1): camera views and corresponding bird's eye views

5.4 The "photography" problem

As regards this issue, we first perform a positioning experiment involving a complex object. We consider a model of the Venus de Milo. In this experiment we first consider a static camera and a moving light. Second, when a minimum of the cost function is reached, we impose motion on the camera. The light must then move in order to maintain a correctly lit statue. The results presented

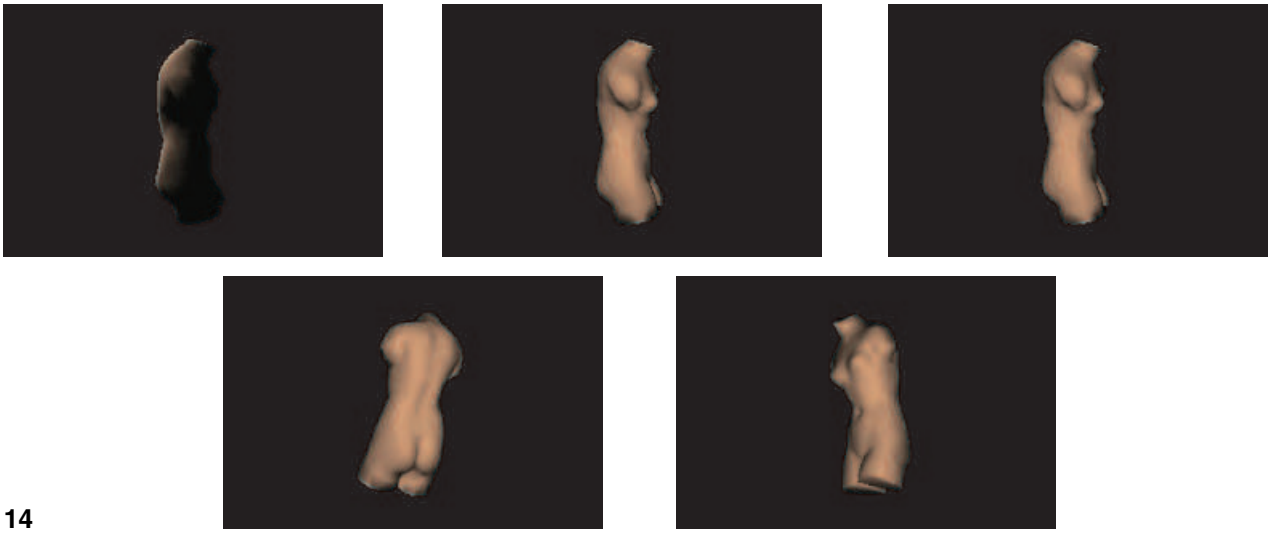


12

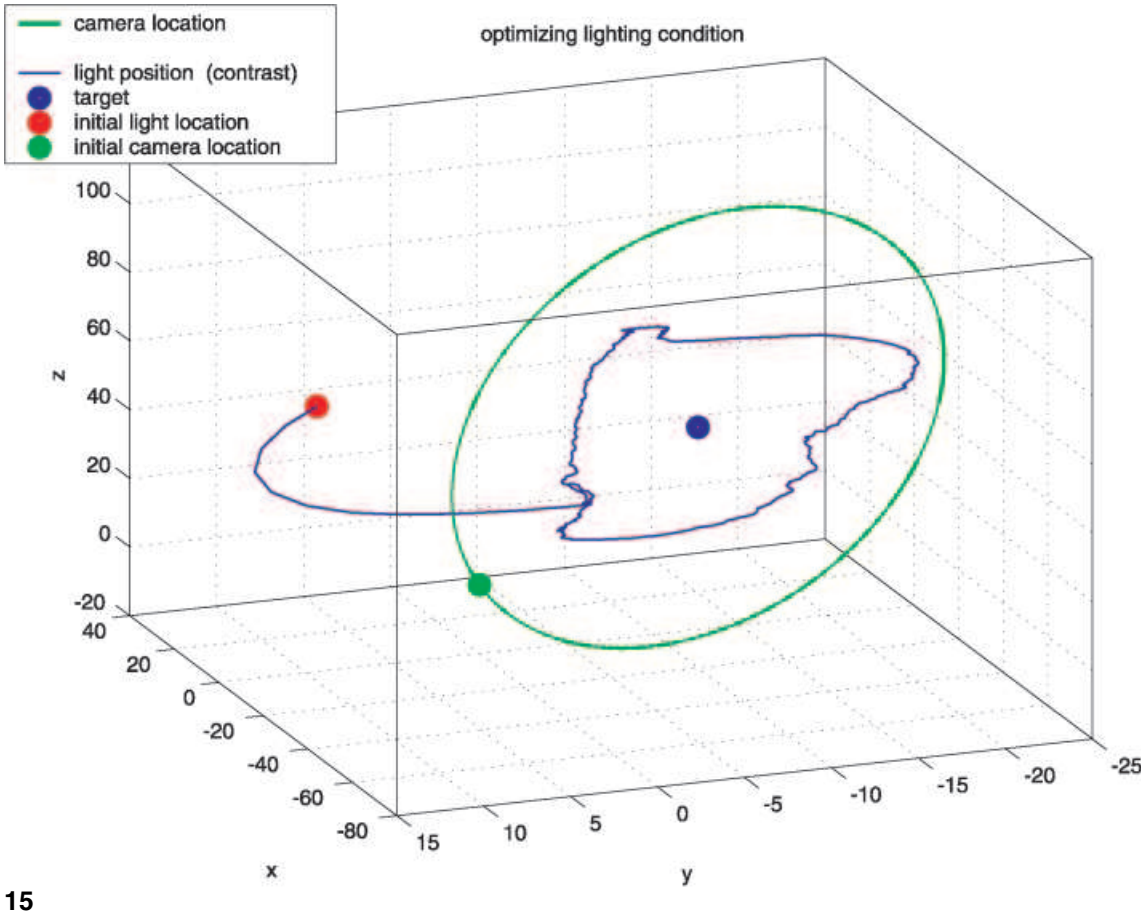


13

Fig. 12. Museum walkthrough (part 2): camera views and corresponding bird's eye views
Fig. 13. Moving in a corridor: bird's eye views and camera views



14



15

Fig. 14. Illuminating the Venus de Milo: maximizing the contrast. In the three first columns the camera remains static, then it turns around the object

Fig. 15. Illuminating the Venus de Milo: camera and light trajectories

(see Fig. 14) show the validity of our approach. One can see that the light trajectories around the statue in Fig. 15.

6 Conclusion

There are many problems associated with the management of a camera in a virtual environment. It is not only necessary to be able to carry out a visual task (often a focusing task or more generally a positioning task) efficiently, but it is also necessary to be able to react in an appropriate and efficient way to modifications of this environment. We chose to use techniques widely considered in the robotic vision community. The basic tool that we considered is visual servoing, which consists of positioning a camera according to the information perceived in the image. This image-based control constitutes the first novelty of our approach. The task is indeed specified *in 2D space*, while the resulting camera trajectories are *in 3D space*. It is thus a very intuitive approach of animation, since it is carried out according to what one wishes to observe in the resulting images sequence.

However, this is not the only advantage of the method. Indeed, in contrast to previous work [7], we did not limit ourselves to positioning tasks with respect to virtual points in static environments. In many applications (such as video games) it is indeed necessary to be able to react to modifications of the environment, of trajectories of mobile objects, etc. We thus considered the introduction of constraints into camera control. Thanks to the redundancy formalism, the secondary tasks (which reflect the constraints on the system) do not have any effect on the visual task. To show the validity of our approach, we have proposed and implemented various classic problems from simple tracking tasks to more complex tasks such as occlusion and obstacle avoidance or positioning with respect to lit aspects of an object (in order to ensure good "photography"). The approach that we proposed has real qualities, and the very encouraging results obtained suggest that the use of visual control for computer animation is a promising technique. The main drawback is a direct counterpart of its principal quality: the control is carried out in the image, thus implying loss of control of the 3D camera trajectory. This 3D trajectory is computed *automatically* to ensure the visual and the secondary tasks but is not controlled by the animator. For this reason, one can undoubtedly see a wider interest in the use of these techniques within real-time reactive applications.

Acknowledgements. The authors wish to thank François Chaumette for valuable comments and Rémi Cozot for submitting the lighting problem to us.

Animations online

Most of the animations presented in this paper can be found as mpeg films on the VISTA group Web page (`http://www.irisa.fr/vista`; follow the "demo" link).

References

1. Arijon D (1976) Grammar of the Film Language. Communication Arts Books, New York
2. Blinn J (1998) Where am I? what am I looking at? IEEE Comput Graph Appl 8:76–81
3. Christianson DB, Anderson SE, He L-W, Salesin DH, Weld DS, Cohen MF (1996) Declarative camera control for automatic cinematography. In: Proc. of AAAI'96 Conference, Portland, Ore., pp 148–155
4. Cowan CK, Kovesi PD (1988) Automatic sensor placement from vision task requirements. IEEE Trans Pattern Anal Mach Intell 10(3):407–416
5. Drucker SM, Zeltzer D (1994) Intelligent camera control in a virtual environment. In: Davis WA, Joe B (eds) Graphics Interface'94, Banff, pp 190–199
6. Espiau B, Chaumette F, Rives P (1992) A new approach to visual servoing in robotics. IEEE Trans Rob Autom 8(3):313–326
7. Gleicher M, Witkin A (1992) Through-the-lens camera control. In: ACM Computer Graphics, SIGGRAPH'92, Chicago, pp 331–340
8. Hashimoto K (1993) Visual Servoing: Real Time Control of Robot Manipulators Based on Visual Sensory Feedback. World Scientific Series in Robotics and Automated Systems, Vol. 7, World Scientific, Singapore
9. He L-W, Cohen MF, Salesin DH (1996) The virtual cinematographer: a paradigm for automatic real-time camera control and directing. In: Proc. of ACM SIGGRAPH'96, in Comput. Graphics Proc., New Orleans, pp 217–224
10. Hutchinson S, Hager G, Corke P (1996) A tutorial on visual servo control. IEEE Trans Rob Autom 12(5):651–670
11. Kyung MH, Kim M-S, Hong S (1995) Through-the-lens camera control with a simple jacobian matrix. In: Davis WA, Prusinkiewicz P (eds) Proc. of Graphics Interface '95, Quebec, pp 171–178
12. Latombe JC (1991) Robot Motion Planning. Kluwer Academic, Dordrecht
13. LaValle M, González-Baños H-H, Becker C, Latombe J-C (1997) Motion strategies for maintaining visibility of a moving target. In: Proc. IEEE Int. Conf. on Robotics and Automation, ICRA '97, Vol. 1, Albuquerque, N. Mex., pp 731–736

14. Li TY, Lien J-M, Chiu S-Y, Yu T-H (1999) Automatically generating virtual guided tours. In: Proc. of Computer Animation 1999, Geneva, ed. by IEEE Comput. Soc., pp 99–106
15. Marchand E, Hager G-D (1998) Dynamic sensor planning in visual servoing. In: IEEE Int. Conf. on Robotics and Automation, Vol. 3, Lueven, pp 1988–1993
16. Nelson B, Khosla PK (1994) Integrating sensor placement and visual tracking strategies. In: IEEE Int. Conf. Robotics and Automation, Vol. 2, San Diego, pp 1351–1356
17. Noma T, Okada N (1992) Automating virtual camera control for computer animation. In: Thalmann N, Thalmann D (eds) Creating and Animating the Virtual (Proc. Computer Animation '92), Springer, Berlin, pp 177–187
18. Samson C, Le Borgne M, Espiau B (1991) Robot Control: the Task Function Approach. Clarendon, Oxford
19. Tarabanis K, Allen PK, Tsai R (1995) A survey of sensor planning in computer vision. IEEE Trans Rob Autom 11(1):86–104
20. Ware C, Osborne S (1990) Exploration and virtual camera control in virtual three dimensional environments. In Proc. '90 Symposium on Interactive 3D Graphics, pp 175–183
21. Weiss LE, Sanderson AC, Neuman CP (1987) Dynamic sensor-based control of robots with visual feedback. IEEE J Rob Autom 3(5):404–417

ÉRIC MARCHAND is currently an INRIA research scientist at IRISA–INRIA Rennes in the Vista project. He received a PhD in Computer Science from Rennes University in 1996 and spend one year as a Postdoctoral Associates in the AI lab of Computer Science Department at Yale University. His research interests include robotics, computer vision, perception strategies and especially the cooperation between perception and action. The considered applications are 3D reconstruction and multi-sensor cooperation. More recently, he studies new application fields such as active vision in micro-robotics, in mobile robotics, in underwater robotics, and in computer animation.

NICOLAS COURTY is currently a PhD candidate in Computer Science in the SIAMES project at IRISA and for France Telecom R&D in Rennes, France. His research interests include visual servoing for computer animation, behaviour modelling, and controlling humanoid avatars. He obtained an engineering degree from INSA Rennes in computer science, and a DEA from the National University of Rennes in 1999.